

```

-----
        .data 0x10000000      # data section starts in
                                # address 0x10000000
aa:     .word 0x0005         # aa is in the 1st data address
bb:     .word 0x000a         # bb is in the following address
cc:     .word ?              # cc is in the 3rd address
#-----
        .globl main
        .text 0x0400000
#-----
main:
        lw    $t1,aa         # $t1 = aa
        lw    $t2,bb         # $t2 = bb
        slt   $t3,$t1,$t2    # if aa<bb $t3=1, else $t3=0
        beq   $t3,$zero,agtb # if $t3=1 goto agtb
        sw    $t2,cc         # else write bb into cc
        j     end
agtb:   sw    $t1,cc         # write aa into cc
end:    j     end
#-----

```

The assembler translates the *lw \$t1,aa* instruction to two consecutive instructions:

```

        lui   $at,0x1000
        lw    $t1,0($at)

```

Preprocessing

- Macro:

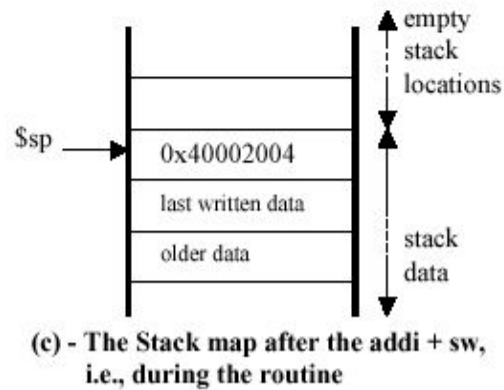
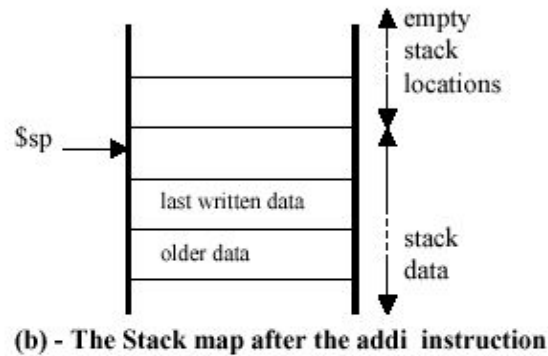
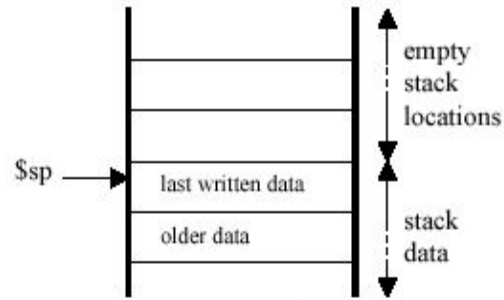
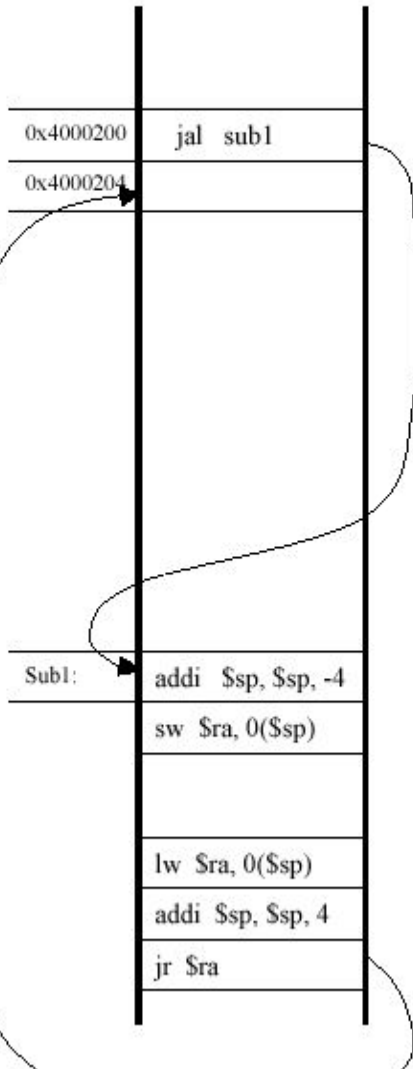
```
.macro      mul3($arg)
add    $t0, $zero, $arg    # save $arg in $t0
add    $arg, $arg, $arg    # double $arg
add    $arg, $t0, $arg     # add $t0
.end_macro
```

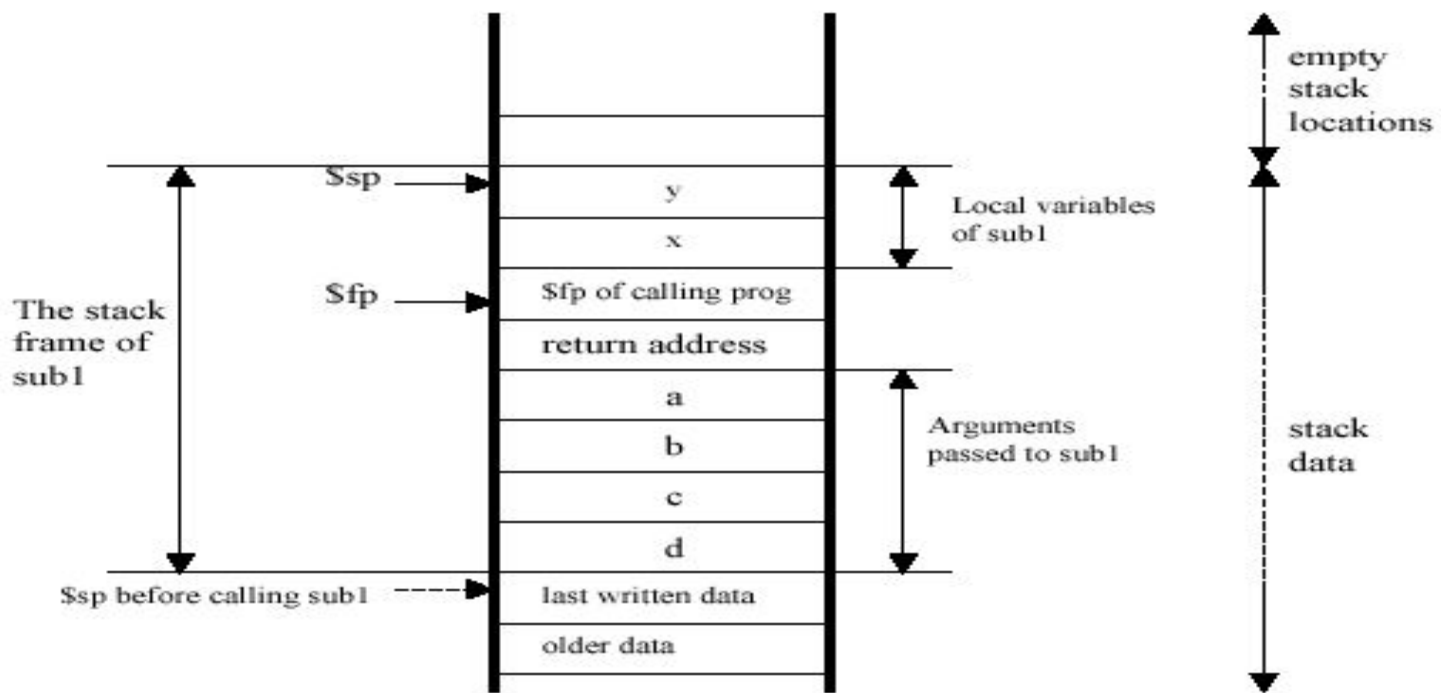
- Code:

```
lui    $at, 0x1000
lw     $s5, 40($at)
mul3($s5)
sw     $s5, 40($at)
```

- After preprocessing

```
lui    $at, 0x1000
lw     $s5, 40($at)
add    $t0, $zero, $s5    # save $arg in $t0
add    $s5, $s5, $s5     # double $arg
add    $s5, $t0, $s5     # add $t0
sw     $s5, 40($at)
```





The routine sub1 has the form of:

```

sub1( int p0, int p1, int p2, int p3, int a, int b, int c, int d)
{
  int x,y;
  -----
  -----
  -----
}

```

The first 4 arguments p0-p3 are transferred through registers \$a0-\$a1

Figure 5.5 - The Stack frame memory map

:Calling procedure

```
addi  $sp, $sp, -16    # prepare room for 4 words in stack
lw     $t1,d           # read d from memory
sw     $t1, 12($sp)    # push d into the stack
lw     $t1,c           #
sw     $t1, 8($sp)     # push c into the stack
lw     $t1,b           #
sw     $t1, 4($sp)     # push b into the stack
lw     $t1,a           #
sw     $t1, 0($sp)    # push a into the stack
jal    sub1            # call sub1
addi  $sp, $sp, 16    # release (pop) the 4 words
```

:Callee procedure

```
:Sub1:  addi    $sp, $sp, -8      # allocate 2 words for the $fp and $ra
        sw     $ra, 4($sp)    # save the $ra
        sw     $fp, 0($sp)    # save the $fp
        add    $fp, $sp, $zero # set the new value for the $fp
        addi   $sp, $sp, -8    # allocate 2 words for the local variables x and y
        -----
        -----
        -----
        -----
        # here we write the body of the routine:

        # In the routine we access x and y by :
        sw     $s3, -4($fp)    # store $s3 in x
        lw     $t5, -8($fp)    # load $t5 with y

        # We access the arguments a,b,c,d by:
        lw     $t1, 8($fp)     # read a from the stack into $t1
        lw     $t2, 12($fp)    # read b from the stack into $t2
        lw     $t3, 16($fp)    # read c from the stack into $t3
        lw     $t4, 20($fp)    # read d from the stack into $t4

        # at the same time, we can use the $sp to spill registers to the stack
        # and still use the $fp to access arguments and local variables
        -----
        -----
        -----
        -----
        # We end the routine by:
        add    $sp, $fp, $zero # release the local variables
        lw     $ra, 4($sp)     # retrieve $ra
        lw     $fp, 0($sp)     # retrieve $fp
        addi   $sp, $sp, 8     # release 2 more words
        jr     $ra             # return to the calling program
```

Single Cycle

